1G−5

大規模アドレス空間を利用するOSの構想 *

申承昊, 瀬川英生, 野末浩志, 岡本利夫, 前田賢一, 斉藤光男
(株)東芝

1 はじめに

現在64ビットアドレス空間を持つCPUが出現しつつある。その巨大なアドレス空間を、従来のOSで行なってきたように各プロセスに一つずつ割り当てるということにはあまり利点はない。その理由は以下の通りである。

- 32ビット以上のアドレス空間を必要とするプログラムは稀であり、全てのプロセスに64ビットのアドレス空間を与える必要はない。[1]
- 分散環境においては、クライアント・サーバ型のプログラミングが主流となっている。そして、現在のOSの設計上の中心的な課題の一つとして、いかにクライアントとサーバの間のメッセージ伝送のコストを下げるかということがあげられている。[2]

そこで、本発表では64ビットアドレス空間を有効に利用することによって、オーバヘッドの小さいデータの共有と、効率的なクライアント・サーバ間の通信を可能にするOSを提案する。

2 メモリモデル

本OSはメモリモデルとして単一仮想記憶と一元化記憶を採用する。単一仮想記憶ということはすなわち全てのプロセスが一つのアドレス空間を共有するということであり、一元化記憶ということは、計算機内の全ての資源はアドレスで指定できるということである。このメモリモデルの利点を、UNIXのメモリモデルと対比して以下に説明する。

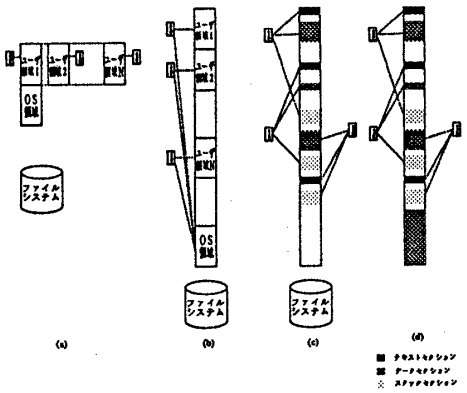

図1: メモリモデルの比較

2.1 多重仮想記憶と単一仮想記憶

UNIXのメモリモデルは図1(a)に示すように、各プロセスがそれぞれ別々の論理アドレス空間の中で実行するというものである。論理アドレス空間内にはプロセスに固有のユーザ領域と全てのプロセスで共有するOS領域が存在する。クライアントとサーバは別々のプロセスであるため、1回のRPCのために2回のコンテキストスイッチが行なわれることになる。

コンテキストスイッチのコストは、

- コンテキストスイッチを行なう命令を実行するためのコスト
- コンテキストスイッチによって引き起こされるキャッシュミスのコスト

に分けられるが、後者の影響の方がより重大であることが分かってきている[2]。特に論理キャッシュを採用した場合は、コンテキストスイッチを行なう度にキャッシュの内容を無効化しなければならないため、そのコストはさらに大きくなるものと考えられる。

そこで、図1(b)のように、全てのプロセスを単一の64ビットアドレス空間内で実行させることを考える。このようにすると、プロセスを切替えても論理アドレスの内容は変わらないため、論理キャッシュをフラッシュする必要が無くなり、コンテキストスイッチのコストが減少する。これが単一仮想記憶を採用することによる効果である。

2.2 アドレス空間とスレッドの分離

図1(b)では、プロセスはユーザ領域とOS領域という2つの領域だけをアクセスしながら実行しているが、この制限を廃し、単一仮想記憶内を自由にアクセス出来るようにしたものが図1(c)である。こうなると、プロセスを動作主体とアドレス空間内の領域に分離して考える必要が出てくる。そこで、以後は動作主体のことをスレッドと呼び、アドレス空間内の領域をメモリセクションと呼ぶ。メモリセクションのうち、スレッドの実行するプログラムテキストが格納されているものをテキストセクション、データが格納されているものをデータセクション、スタックが格納されているものをスタックセクション呼ぶことにする。

このメモリモデルでは、一つのスレッドが複数の独立したプログラムを実行することが出来る。RPCのクライアントは、スレッドがクライアントプログラムを実行している状態であり、サーバは、同じスレッドがサーバプログラムを実行している状態のことになる。このため、従来のRPCがコンテキストスイッチなしで行なえるようになる。

2.3 一元化記憶

さらに、図1(d)のように、一元化記憶を採用すると、従来ファイルとしてアクセスしていたデータを、アドレス指定によってアクセスすることが可能になる。このメモリモデルでは、データアクセスの際にOSの処理が不要となるため、データアクセス時間が短縮される。また、プログラムを新たに起動する時には、ファイルからメモリ上にロードする必要がなく、アドレス空間内に存在しているそのプログラムのイメージをそのまま実行することが出来る。これにより、プログラムの起動時間も短縮される。

3 メモリプロテクション

アドレス空間から全ての計算機資源がアクセス出来るというメモリモデルを採用するOSでは、従来のOSにおいてファイルシステムやアドレス空間の多重化によって実現されてきたデータのプロテクションの機能の全てを、メモリのプロテクションの機能を使って実現しなければならない。そこで、我々は新たに開発する高機能のMMUを利用することに

*Concept of 64bit-address space oriented OS. Sung Ho SHIN, Hideo SEGAWA, Hiroshi NOZUE, Toshio OKAMOTO, Ken'ichi MAEDA, Mitsuo SAITOH. Toshiba Corporation.

よって、メモリセクション単位のきめの細かいメモリプロテクションを行なう [3]。

このMMUの扱うページテーブルの各エントリには、図2に示すように、アクセス・コントロール・リスト (ACL) を付加する。

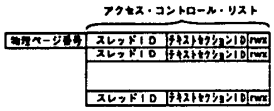

図 2: ページテーブルエントリ

ACLの設定によって、特定のプログラムからのみアクセス出来るデータ・セクションや、あるスレッドからのみアクセス可能なデータ・セクションを作ることが出来る。また、RPCを行なう時に、サーバプログラムのエントリポイントを限定することが可能となる。

3.1 RPCの手順

クライアントプログラムを実行していたスレッドがサーバプログラムを実行する時に、サーバプログラムへのエントリポイントを限定したい場合は、図3のようにACLを設定する。

図の○はアクセスが可能であることを示しており、×はアクセスが禁止されていることを示している。

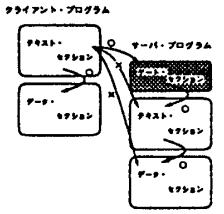

図 3: RPCのエントリポイントの限定

クライアントプログラムを実行しているスレッドがサーバプログラムに移る時には、ゲート・セクションを一旦経由しなければならないようにする。ゲート・セクションにはサーバプログラムの正しいエントリポイントへのジャンプ命令だけが並べられているため、これを実行すると、サーバプログラムの正しいエントリポイントにだけジャンプが行なわれることになる。

3.2 データアクセス

あるデータセクションを、プログラムから直接アドレス指定することでアクセス出来るようにACLを変更する場合には、プロテクション・サーバと呼ばれるサーバを使用する。プロテクション・サーバは、ページテーブルにアクセスすることが出来るプログラムである。このサーバはアクセス権変更要求の内容をチェックし、正当だと判明したときに要求されたACLの書き換えを行ない、要求のあったプログラムからデータ・セクションのアクセスが可能になるようにする。

4 マイクロカーネル構造

上で述べたメモリモデルを採用することにより、本OSが管理する計算機資源は、

- スレッド
- メモリセクション

だけになる。これ以上の機能はサーバプログラムによって提供する。本OSはクライアント・サーバ型の計算が速いことが特長となっているため、OSをモジュール化し、RPCを多用する構成にしても性能に悪影響を与えない。このように本OSは、より高機能なOSのための、マイクロカーネルとして使う。

図4に示すように、本OSをマイクロカーネルとして使用し、この上にポートサーバ、メッセージサーバを実行させることによりMachマイクロカーネルインタフェースを実現する。

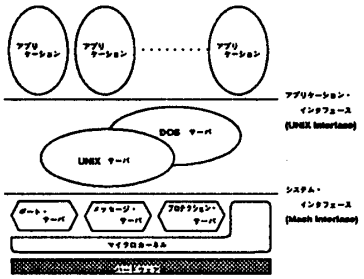

図 4: マイクロカーネル構造

5 おわりに

以上、64ビットCPUの特徴を生かし、分散環境において主流であるクライアント・サーバ型の処理を高速化するOSを提案した。このOSは、単一仮想記憶と一元化記憶を採用し、全ての計算機資源を一つの64ビットアドレス空間内に置く。これにより、クライアントからサーバへのRPCは、サーバプログラムへのサブルーチンコールに置き換えることが出来るため、コンテキストスイッチによるオーバヘッドが無くなる。

また、一元化記憶を採用することで、全てのデータをアドレスによって高速にアクセスすることが可能となる。

その他にも本OSは、以下の特長を持つ。

- マイクロカーネル構造
- Machマイクロカーネル・インタフェースをサポート

本OSで採用するメモリモデルにおいて、データの保護を行なうためには高機能のMMUが必要であり、現在はこのMMUの設計中である。また、これに並行して80386のセグメントの機能を使った、本メモリモデルの実現に着手したところである。

今後はネットワークで接続された複数の計算機の間で、単一仮想空間をネットワーク透過に共有させる方法についても研究をすすめてゆく。

参考文献

[1] Jhon R. Mashey, "64-bit Computing," Byte, pp.135–142, September 1991.

[2] Brian N. Bershad, Thomas Anderson, Edward Lazowska and Henry Levy, "User-Level Interprocess Communication for Shared Memory Multiprocessors," ACM Transactions on Computer Systems, Vol.9, No.2, May 1991, pp.175–198.

[3] 野末浩志, 斉藤光男, "大規模アドレス空間内でのデータ保護方式の実現", 情報処理学会第44回全国大会予稿集,Mar 1992(予定).